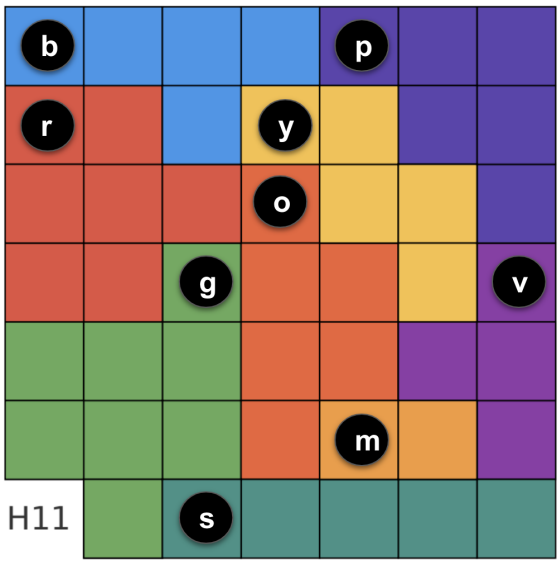
CSCE 625 HW2 Report

Tianqi Li

February 28, 2019

1 Task 1

Use your program to find a solution for each of the thirteen hole locations. (An example solution one of the holes is shown in Figure 2.)



(a) One solution to the assembling question

(b) Flatten the matrix to a list

(c) Convert the solution to list

Figure 1: The representation of the searching state, a) 9 pieces are placed to meet the constraint of the 7 × 7 board; b) regard this board as a 7 × 7 matrix, we can flatten the matrix into a list, and for each piece, we can locate them by the index of first cell appear in the list, piece **b** is at index = 0, piece **p** is at index = 4, so on; c) from b, we can make the presentation of the puzzle much easier by just using the sequence of the piece in b), which makes a list with length of 9.

Answer. First step of finding the solution is presenting the problem in a proper method. This geometry problem in a 2 dimension grid space is somewhat tricky for computer programs to handle because usually variables and data structures in program are in 1 dimension, like queue, stack, tree, etc. Based on the representation of digital image, shown in Fig 1, I came up with the idea of present

the problem of fitting 9 pieces into a board as finding the sequence of 9 different variables in a list.

This problem has 9 pieces, with each piece can be rotated 90° in one step, leads to 4 different shapes (or we call it 4 dimensions for each variable). For block **m** and **s** in Fig 1 a, each of them has 2 dimensions, vertical and horizontal. From page 214 in [1], for a CSP with n variables of domain size d, the searching tree will have $n!d^n$ leaves, in such case we will have to search $9! \times 4^8 = 23,781,703,680$ different situations! How to relax the search should be solved.

Given the presentation, both DFS and BFS guarantee to return the solution for us. But with the aforementioned 23-billion-leave-tree to search, I choose DFS to save memory for my mac. Due to BFS will maintain the expanded tree nodes much more than DFS.

From the discussion in class, we tried to make some "shortcut" like start from the corner or start from the big pieces. Here I tried is to start from big pieces whose sizes are defined as the sum of their width and length. Trying from big pieces will easily leads to the termination due to the constraint of width of the board when big pieces meet in same row, which follows the principle of minimum-remaining value (MRV) heuristic mentioned in Section 6.3.1 in [1].

Based on the performance comparison of direct DFS and DFS with piece shape heuristic, a direct DFS requires more than 20 min (or more than my patience) to finish DFS, but the DFS with piece shape heuristic only uses less than 10 min to find (hopefully all) solutions 13 puzzles.

The pesudocode are presented as Algorithm 1 SearchSolutions, which is based on DFS but using the sorted pieceList to expand search from big pieces to small ones. The main script to trigger the search is `task1.py`.

In Fig 2 provides one solution to each puzzles.

Algorithm 1 SearchSulutions

```
pieceList = [g,r,b,...];
board = a 0 matrix of 7 × 7;
set the hole position to 1;
pieceList = sortPiece(pieceList);
DFS(pieceList, board)  use DFS to search on the board by piece in the sorted order
haha
```

Algorithm 2 sortPiece

```
# This sort is to order the
LaTeXsortPiece(pieceList)
for piece in pieceList do
  piece.size = piece.width + piece.length
end for
pieceList = sort pieceList by piece.size descending
return  pieceList
```

2 Task 2

For each of the thirteen hole locations, compute an estimate of the total number of distinct solutions there are.

Answer. From the solution provided by DFS, the result are shown in Table 1. Additionally, I make a solution number distribution of each hole location on the board, as Fig 3 shows.

Table 1: Collected result from DFS

Problem	1	2	3	4	5	6	7
Number of solutions	38	92	45	26	50	12	43
Problem	8	9	10	11	12	13	
Number of solutions	37	18	70	181	17	67	

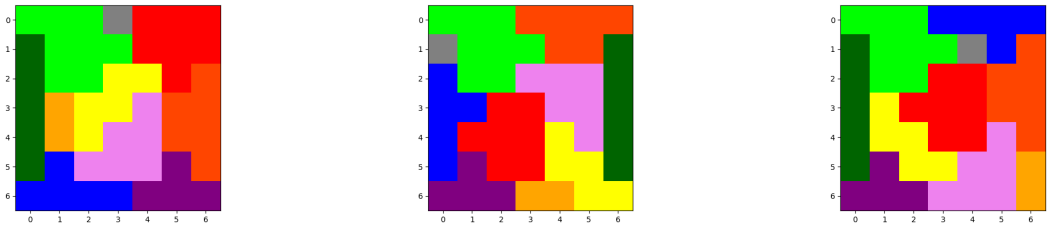
3 Task 3

Are there locations other than {H1, H2, . . . , H13} where a hole might be placed to give a new puzzle?

Answer. No, there are no other new hole for new puzzle. Taken the rule that the pieces are allowed to rotate into consideration, we can actually rotate the holes around the centre cell (cell at (4,4) of the board) with step of 90° which will provide the same puzzle. For each hole we rotate 4 steps to generate 4 holes representing the same puzzle, and if we rotate all 13 holes, we will found that all cells on the board are covered by holes. Thus there are no other chance to make a new puzzle given the holes set {H1, H2, . . . , H13}.

References

[1] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.



(a) H1 (b) H2 (c) H3

(d) H4 (e) H5 (f) H6

(g) H7 (h) H8 (i) H9

(j) H10 (k) H11 (l) H12

(m) H13

Figure 2: One solution to each hole

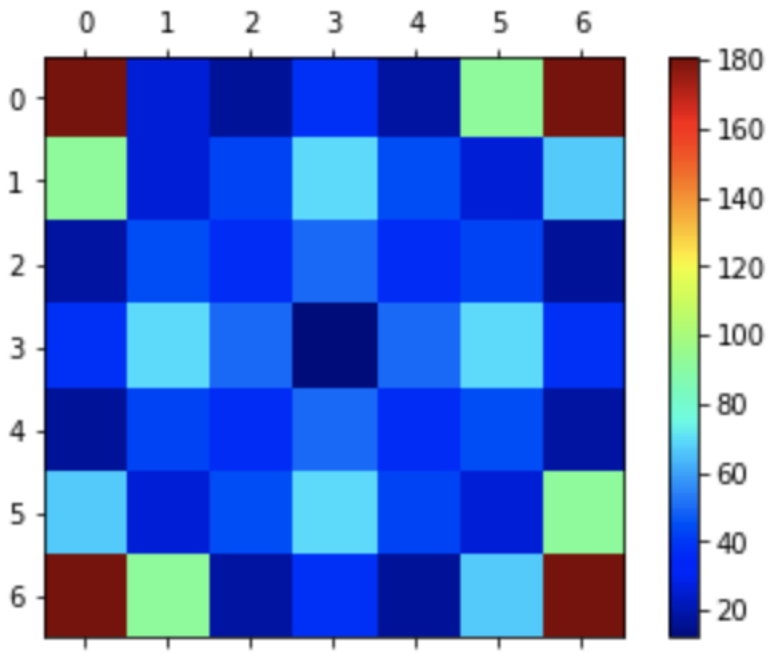


Figure 3: This is the solution number distribution by holes. The colorbar maps the color in each hole with its solution found by DFS.